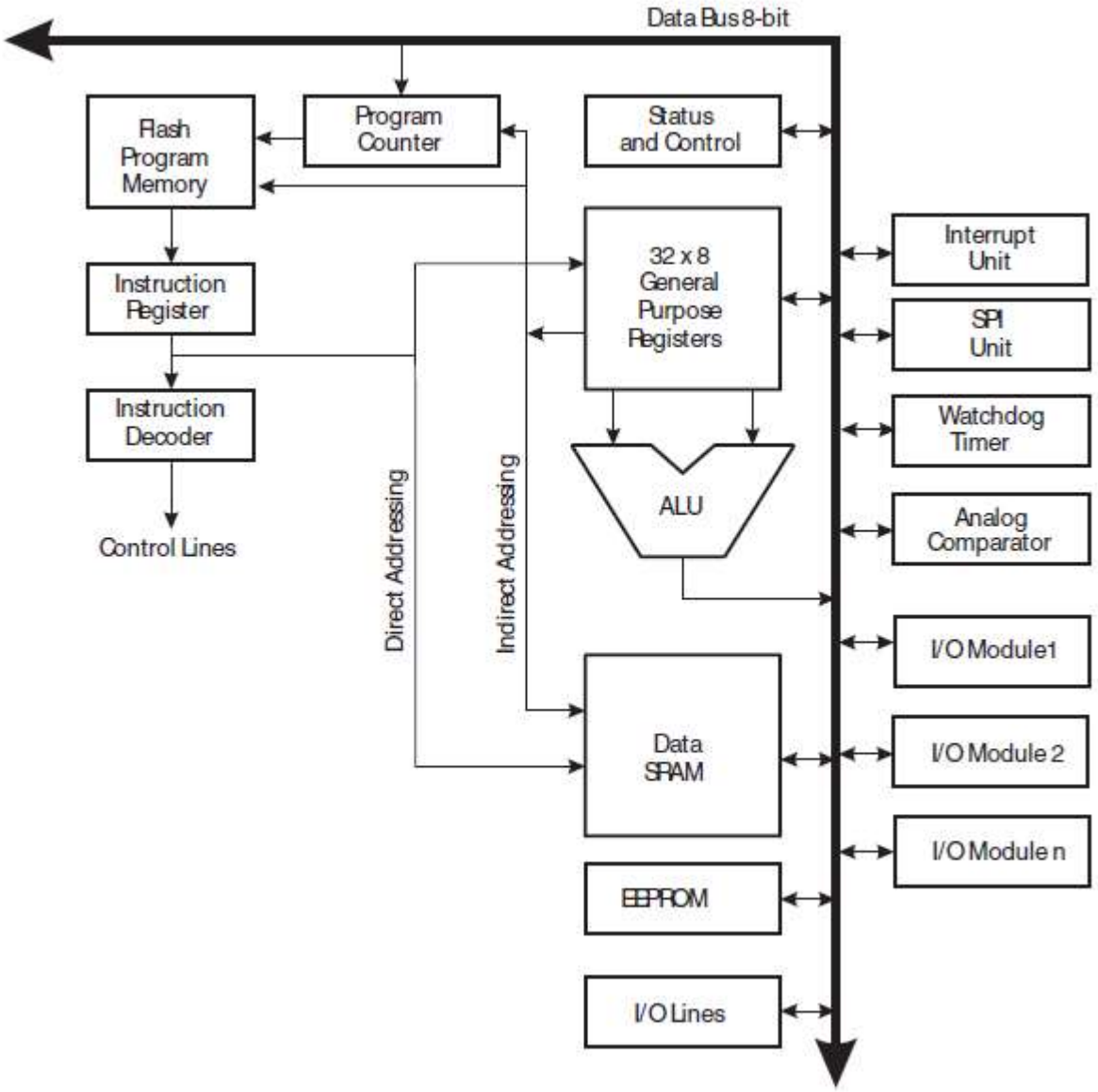
# Núcleo (AVR CPU Core)

Sistemas Microcontrolados


Data Bus 8-bit
Flash Program Memory
Program Counter
Status and Control
Instruction Register
Instruction Decoder
Control Lines
32 x 8 General Purpose Registers
ALU
Direct Addressing
Indirect Addressing
Data SRAM
EEPROM
I/O Lines
Interrupt Unit
SPI Unit
Watchdog Timer
Analog Comparator
I/O Module1
I/O Module 2
I/O Module n

# ALU – Arithmetic Logic Unit

| Mnemonics | Operands | Description | Operation | Flags | #Clocks |
|---|---|---|---|---|---|
| ARITHMETIC AND LOGIC INSTRUCTIONS | | | | | |
| ADD | Rd, Rr | Add two Registers | Rd ← Rd + Rr | Z, C, N, V, H | 1 |
| ADC | Rd, Rr | Add with Carry two Registers | Rd ← Rd + Rr + C | Z, C, N, V, H | 1 |
| ADIW | Rdl,K | Add Immediate to Word | Rdh:Rdl ← Rdh:Rdl + K | Z, C, N, V, S | 2 |
| SUB | Rd, Rr | Subtract two Registers | Rd ← Rd - Rr | Z, C, N, V, H | 1 |
| SUBI | Rd, K | Subtract Constant from Register | Rd ← Rd - K | Z, C, N, V, H | 1 |
| SBC | Rd, Rr | Subtract with Carry two Registers | Rd ← Rd - Rr - C | Z, C, N, V, H | 1 |
| SBCI | Rd, K | Subtract with Carry Constant from Reg. | Rd ← Rd - K - C | Z, C, N, V, H | 1 |
| SBIW | Rdl,K | Subtract Immediate from Word | Rdh:Rdl ← Rdh:Rdl - K | Z, C, N, V, S | 2 |
| AND | Rd, Rr | Logical AND Registers | Rd ← Rd • Rr | Z, N, V | 1 |
| ANDI | Rd, K | Logical AND Register and Constant | Rd ← Rd • K | Z, N, V | 1 |
| OR | Rd, Rr | Logical OR Registers | Rd ← Rd v Rr | Z, N, V | 1 |
| ORI | Rd, K | Logical OR Register and Constant | Rd ← Rd v K | Z, N, V | 1 |
| EOR | Rd, Rr | Exclusive OR Registers | Rd ← Rd ⊕ Rr | Z, N, V | 1 |
| COM | Rd | One's Complement | Rd ← 0xFF – Rd | Z, C, N, V | 1 |
| NEG | Rd | Two's Complement | Rd ← 0x00 – Rd | Z, C, N, V, H | 1 |
| SBR | Rd,K | Set Bit(s) in Register | Rd ← Rd v K | Z, N, V | 1 |
| CBR | Rd,K | Clear Bit(s) in Register | Rd ← Rd • (0xFF - K) | Z, N, V | 1 |
| INC | Rd | Increment | Rd ← Rd + 1 | Z, N, V | 1 |
| DEC | Rd | Decrement | Rd ← Rd – 1 | Z, N, V | 1 |
| TST | Rd | Test for Zero or Minus | Rd ← Rd • Rd | Z, N, V | 1 |

# ALU – Arithmetic Logic Unit

| | | | | | |
|---|---|---|---|---|---|
| CLR | Rd | Clear Register | Rd ← Rd ⊕ Rd | Z, N, V | 1 |
| SER | Rd | Set Register | Rd ← 0xFF | None | 1 |
| MUL | Rd, Rr | Multiply Unsigned | R1:R0 ← Rd x Rr | Z, C | 2 |
| MULS | Rd, Rr | Multiply Signed | R1:R0 ← Rd x Rr | Z, C | 2 |
| MULSU | Rd, Rr | Multiply Signed with Unsigned | R1:R0 ← Rd x Rr | Z, C | 2 |
| FMUL | Rd, Rr | Fractional Multiply Unsigned | R1:R0 ← (Rd x Rr) << 1 | Z, C | 2 |
| FMULS | Rd, Rr | Fractional Multiply Signed | R1:R0 ← (Rd x Rr) << 1 | Z, C | 2 |
| FMULSU | Rd, Rr | Fractional Multiply Signed with Unsigned | R1:R0 ← (Rd x Rr) << 1 | Z, C | 2 |
| **BRANCH INSTRUCTIONS** | | | | | |
| RJMP | k | Relative Jump | PC ← PC + k + 1 | None | 2 |
| IJMP | | Indirect Jump to (Z) | PC ← Z | None | 2 |
| EIJMP | | Extended Indirect Jump to (Z) | PC ←(EIND:Z) | None | 2 |
| JMP | k | Direct Jump | PC ← k | None | 3 |
| RCALL | k | Relative Subroutine Call | PC ← PC + k + 1 | None | 4 |
| ICALL | | Indirect Call to (Z) | PC ← Z | None | 4 |
| EICALL | | Extended Indirect Call to (Z) | PC ←(EIND:Z) | None | 4 |
| CALL | k | Direct Subroutine Call | PC ← k | None | 5 |
| RET | | Subroutine Return | PC ← STACK | None | 5 |
| RETI | | Interrupt Return | PC ← STACK | I | 5 |
| CPSE | Rd,Rr | Compare, Skip if Equal | if (Rd = Rr) PC ← PC + 2 or 3 | None | 1/2/3 |
| CP | Rd,Rr | Compare | Rd – Rr | Z, N, V, C, H | 1 |
| CPC | Rd,Rr | Compare with Carry | Rd – Rr – C | Z, N, V, C, H | 1 |

# Status Register

- O *Status Register* contém informações sobre os resultados de operações aritméticas mais recentes.
- Estas informações podem ser usadas para alterar o fluxo de programa para atender instruções condicionais.
- Portanto o *Status Register* é atualizado após cada operação da ALU
- Em muitos casos exclui-se a necessidade de utilizar instruções de comparação dedicadas resultando em maior velocidade e código mais compacto.

### 7.4.1 SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x3F (0x5F) | I | T | H | S | V | N | Z | C | SREG |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the “Instruction Set Summary” on page 404.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

| | | | | | |
|---|---|---|---|---|---|
| BST | Rr, b | Bit Store from Register to T | T ← Rr(b) | T | 1 |
| BLD | Rd, b | Bit load from T to Register | Rd(b) ← T | None | 1 |

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the "Instruction Set Summary" on page 404 for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the "Instruction Set Summary" on page 404 for detailed information.

- **Bit 3 – V: Two's Complement Overflow Flag**

The Two's Complement Overflow Flag V supports two's complement arithmetics. See the "Instruction Set Summary" on page 404 for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the "Instruction Set Summary" on page 404 for detailed information.

- **Bit 1 – Z: Zero Flag**

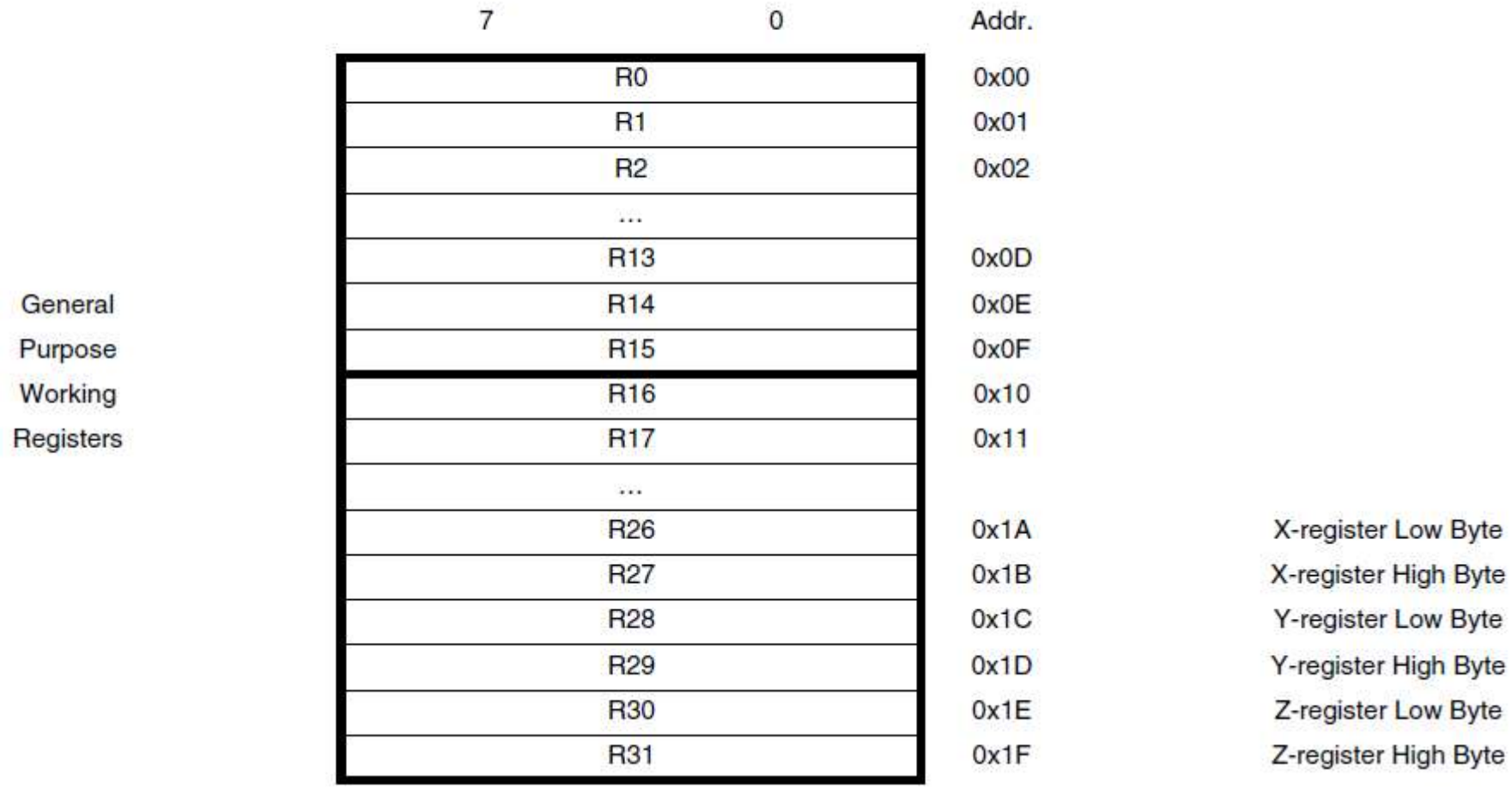
The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the "Instruction Set Summary" on page 404 for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the "Instruction Set Summary" on page 404 for detailed information.

# General Purpose Register File

**Figure 7-2.** AVR CPU General Purpose Working Registers



# Stack Pointer

- A Pilha “Stack”  e usado principalmente para armazenar dados temporariamente, variáveis locais e endereço de retorno após interrupções e chamadas de sub-rotinas.
- O registrador de ponteiro de pilha “Stack Pointer Register” sempre aponta para o topo da pilha.
- Note que a pilha é implementada de forma decrescente, de posições de memória maiores para menores. Isso significa que o comando “Stack PUSH” decrementa o ponteiro de memória.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x3E (0x5E) | **SP15** | **SP14** | **SP13** | **SP12** | **SP11** | **SP10** | **SP9** | **SP8** | **SPH** |
| 0x3D (0x5D) | **SP7** | **SP6** | **SP5** | **SP4** | **SP3** | **SP2** | **SP1** | **SP0** | **SPL** |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

# Stack Pointer

- O stack pointer deve estar configurado para o endereço 0x0200.
- O endereço inicial da Pilha é o ultimo endereço da SRAM interna.
- O Stack pointer é decrementado quando dados são empurrados na pilha pela instrução PUSH e incrementado quando os dados forem retirado com a instrução POP.